⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　昭62-283496

| ⑤1Int.Cl.⁴ | 識別記号 | 庁内整理番号 | ⑬公開　昭和62年(1987)12月9日 |
|---|---|---|---|
| G 11 C 17/00 | 307 | 6549-5B | |

審査請求　未請求　発明の数 1　(全9頁)

⑤4発明の名称　プログラマブルリードオンリメモリの書き込み回数管理方式

⑳特　願　昭61-124731

㉒出　願　昭61(1986)5月31日

㉗発明者　仲田　眞一　東京都大田区下丸子3丁目30番2号　キヤノン株式会社内

㉑出願人　キヤノン株式会社　東京都大田区下丸子3丁目30番2号

㉔代理人　弁理士 小林　将高

明　細　書

1．発明の名称

プログラマブルリードオンリメモリの書き込み回数管理方式

2．特許請求の範囲

記憶領域に書き込まれた情報を電気的に消去可能なプログラマブルリードオンリメモリにおいて、前記記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込むとともに、前記書き込み頻度の低いブロックを未使用ブロックの最後尾に接続させることを特徴とするプログラマブルリードオンリメモリの書き込み回数管理方式。

3．発明の詳細な説明

〔産業上の利用分野〕

この発明は、電気的消去可能なプログラマブルリードオンリメモリの書き込み回数の管理方式に関するものである。

〔従来の技術〕

従来のＥＥＰＲＯＭ（Electrical Eresable and Programmable ROM）は、容量も少なく、また書き込むために必要な外部回路が多かった。さらに、チップ内のすべてのデータを消去するモードしか有していなかった。最近は、容量も大きくなるとともに、外部回路も殆ど必要なくＣＰＵのアドレスバス，データバスに結線できるようになり、またＥＥＰＲＯＭ内の１バイトのデータのみの消去も可能となってきた。以上の改良により、使用目的によっては、従来のランダムアクセスメモリ（ＲＡＭ）で構成していた機能の置換が可能となった。

例えば、従来の小型パソコン、日本語ワープロで作成したプログラムや文章，外字等を保存しておくためにメモリカードと云うものがある。これは、必要なときにパソコン，日本語ワープロ等の本体に差し込んでプログラムや文章を記憶させ、本体から引き抜いても、そのデータを記憶しているように、メモリカード内にはＲＡＭと電池が搭

されていた。そこで、メモリカードをＥＥＰＲＯＭで構成することにより、電池を無くすることができると考えられた。

〔発明が解決しようとする問題点〕

ところが、ＥＥＰＲＯＭでは従来のＲＡＭのように自由に何度も書き換えられない制約があり、すなわち、あらかじめ設定される書き込み回数を越えて、メモリカードへの書き込みを行うことにより、記憶しているはずのデータを消失させてしまう等の問題点があった。またＥＥＰＲＯＭに書き込まれたデータのうち、頻繁に書き換えられるデータと書き換え頻度の少ないデータとが存在し、書き換え頻度の高いデータの書き換え回数が所定値を越えると、ＥＥＰＲＯＭへの書き換えが可能にも関わらず書き換え不能となる問題点があった。

この発明は、上記の問題点を解消するためになされたもので、ＥＥＰＲＯＭに書き込まれるデータの消失を防止するとともに、ＥＥＰＲＯＭへの書き込み回数を平均化させるとともに、ＥＥＰＲＯＭ上の書き換え頻度を平均化して、ＥＥＰＲＯＭへの書き換え寿命を延命できるプログラマブルリードオンリメモリの書き込み回数管理方式を得ることを目的とする。

〔問題点を解決するための手段〕

この発明に係るプログラマブルリードオンリメモリの書き込み回数管理方式は、記憶領域を複数のブロックに分割し、各ブロック毎に書き込み回数を記憶し、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込むとともに、前記書き込み頻度の低いブロックを未使用ブロックの最後尾に接続させる。

〔作用〕

この発明においては、記憶領域の各ブロック毎の書き込み回数を記憶しておき、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込ませるとともに、書き込み頻度の低いブロックを未使用のブロックの最後端に接続させる。

〔実施例〕

第１図（ａ）はこの発明の一実施例を示すプログラマブルリードオンリメモリへの書き込み回数管理方式を説明する模式図であり、１はＥＥＰＲＯＭで、例えば書き込み容量が32768 バイト×8 ビットで、書き込み回数が１万回に設定してある。ＥＥＰＲＯＭ１は、ポインタブロック１ａおよび予備ポインタブロックＳＰＢ１～ＳＰＢ５０より構成される。ポインタブロック１ａは４アドレス（各１バイト）で構成され、『０～１』番地の２バイトで、書き換え回数ＷＣＮＴ、例えば『１３８８16』を記憶している。またポインタブロック１ａの『２』番地の１バイトは、ディレクトリＤＢ、例えば『０１16』を記憶している。さらに、ポインタブロック１ａの『３』番地の１バイトは、未使用のスタートブロック番号ＯＳＢ、例えば『３３16』を記憶している。またポインタブロック１ａの『４』番地の１バイトは、未使用のエンドブロック番号ＯＥＢ、例えば『８Ａ16』を記憶している。

第１図（ｂ）はこの発明の装置構成の一例を説明するブロック図であり、１１はＣＰＵで、ＲＯＭ１１ａ、ＲＡＭ１１ｂを有し、ＲＯＭ１１ａに格納された第７図、第８図に示すフローに応じたプログラムに応じて各部を制御する。１２は入力手段で、データ書き込み装置１３にセットされるＥＥＰＲＯＭ１へのデータ書き込みおよびデータ消去を指示する。なお、ＣＰＵ１１にはデータの転送を行うアキュムレータＡＣＣ、ＢＣＣを有している。

第２図は第１図（ａ）に示すＥＥＰＲＯＭ１の構造を示す模式図であり、２１はブロック番号であり、例えば１２７個のブロックＢＬＯＣＫ１～ＢＬＯＣＫ１２７に分割されている。各ブロックは、例えば２５６バイトで構成され、先頭の２バイトで、そのブロックが更新された回数、すなわち、後述する更新回数が記憶されている。次に続く２５３バイトは記憶データＤＡＴＡが記憶されており、最後の１バイトは、記憶データＤＡＴＡがこのブロックに収まるか、または他のブロックに及ぶかどうかを示す継続ブロックエリアＣＢが

あり、他のブロックに記憶データＤＡＴＡが及ぶ場合は、連続ブロックエリアＣＢには連続するブロック番号が記憶され、他のブロックに記憶データＤＡＴＡが及ばない場合は、連続ブロックエリアＣＢには『ＦＦ$_{16}$』が記憶されている。

第３図は第２図に示す各ディレクトリブロック構造を説明する模式図であり、３０は前記ディレクトリＤＢに指示されるディレクトリブロック、３１は前記ディレクトリブロック３０の更新カウンタで、例えば２バイトで構成される。３２はファイル領域で、各ファイル名が１２バイトで記憶される。３３はスタートブロック番号エリア（ＳＢ）で、例えば１バイトで構成され、ファイルのスタートブロック番号が記憶されている。３４はエンドブロック番号エリア（ＥＢ）で、例えば１バイトで構成され、ファイルのエンドブロック番号が記憶されている。３５はチェーンブロックエリア（ＣＢ）で、ディレクトリブロック３０に連続するディレクトリブロックの有無を記憶する。例えばチェーンブロックエリア３５が『ＦＦ$_{16}$』となる。なお、ディレクトリブロック３０は、例えば１８個のファイル領域３２で構成される。

次に第１図（ａ）および第３図を参照しながらＥＥＰＲＯＭ１の構造について説明する。

第１図（ａ）に示すようにポインタブロック１ａの書き換え回数ＷＣＮＴに、例えば『１３８８$_{16}$』が記憶されているとすると、５０００回の更新が行われたことを示し、またディレクトリＤＢには『０１$_{16}$』が記憶されているので、ディレクトリＤＢに指示されるディレクトリブロック３０のブロック番号が『１』で、そのディレクトリブロック３０の更新カウンタ３１には、『１４２Ｆ$_{16}$』が記憶されている。これは、このディレクトリブロック３０を５１６７回更新したことを示し、ファイル領域３２のファイル（ＦｉＬｅ）１（ファイル名）はスタートブロック番号エリア３３が『０２$_{16}$』で、エンドブロック番号エリア３４が『０５$_{16}$』となっているため、ブロックＢＬＯＣＫ２から始まり、ブロックＢＬＯＣＫ５で終ることになる。またファイル領域３２のファイル２は、スタートブロック番号エリア３３が『０Ａ$_{16}$』で、エンドブロック番号エリア３４が『０Ｆ$_{16}$』となっているため、ブロックＢＬＯＣＫ１０から始まり、ブロックＢＬＯＣＫ１５で終ることになる。さらに、ファイル領域３２のファイル３（ファイル名）は、スタートブロック番号エリア３３が『１５$_{16}$』で、エンドブロック番号エリア３４が『１８$_{16}$』となっているため、ブロックＢＬＯＣＫ２１から始まり、ブロックＢＬＯＣＫ２４で終ることになる。またファイル領域３３のファイル３の次に『ＦＦ$_{16}$』が書かれているので、このファイル領域３３はファイル３で終了していることになる。

第４図は未使用のＥＥＰＲＯＭ１の状態を説明する模式図であり、第１図（ａ），第３図と同一のものには同じ符号を付している。

この図から分かるように、未使用のＥＥＰＲＯＭ１のポインタブロック１ａの書き換え回数ＷＣＮＴが『０００１$_{16}$』、ディレクトリＤＢが『０１$_{16}$』、未使用のスタートブロック番号ＯＳＢが『０２$_{16}$』、未使用のエンドブロック番号ＯＥＢが『７Ａ$_{16}$』がそれぞれポインタブロック１ａの０番地から４番地にそれぞれ記憶されている。これにより、ディレクトリＤＢに指示されるブロックＢＬＯＣＫ１を参照すると、更新カウンタ３１に『０００１$_{16}$』が書き込まれているとともに、ファイル領域３２のファイル１に『ＦＦ$_{16}$』が書き込まれており、さらに、チェーンブロックエリア３５に『ＦＦ$_{16}$』が書き込まれており、ＥＥＰＲＯＭ１が未使用状態であることを示している。

さらに、ポインタブロック１ａのスタートブロック番号ＯＳＢおよびエンドブロック番号ＯＥＢには『０２$_{16}$』，『７Ｆ$_{16}$』がそれぞれ書き込まれている。すなわち、ブロックＢＬＯＣＫ２～１２７には先頭の２バイトに『０００１$_{16}$』が書き込まれ、最終の１バイトに各後続のブロックの連続を示すチェーンブロックエリア３５には、ブロックＢＬＯＣＫ２～１２６に対して『０３～７Ｆ$_{16}$』が書き込まれ、ブロックＢＬＯＣＫ１２７のチェーンブロックエリア３５には『ＦＦ』が書

込まれている。このように、各ブロックＢＬＯＫ２～１２７は１つのチェイン構造となる。

次に第３図，第５図（ａ），（ｂ）を参照しながらＥＥＰＲＯＭ１への書き込み動作を説明する。

第５図（ａ），（ｂ）はＥＥＰＲＯＭ１への書き込み動作を説明する模式図であり、第１図（ａ），第３図と同一のものには同じ符号を付している。なお、書き込み直前は、第３図に示す状態であったものとする。

まず、各ブロックＢＬＯＣＫのファイル領域３２の先頭が『００₁₆』のところを探し当てる。第３図の場合は、ファイル２とファイル３との間に『００₁₆』があり、そこにファイル４という名前を１２バイトで書き込み、ポインタブロック１ａの未使用ブロックのスタートブロック番号ＯＳＢを参照して、スタートブロック番号ＯＳＢの指示するブロックＢＬＯＣＫ、すなわち『５７₁₆』の先頭の２バイト情報、すなわち、更新カウンタ３１を『１』インクリメントし、その加算値が、例えば１万回を越えているようであれば、ファイル４のチェーンブロックエリア３５が示すブロックＢＬＯＣＫに対して同様の操作を行い、更新カウンタ３１が１万回以下のブロックＢＬＯＣＫを探し当てて、そのブロックＢＬＯＣＫの番号をポインタブロック１ａのスタートブロック番号ＯＳＢに書き込むとともに、ファイル４のデータをブロックＢＬＯＣＫ８７（２５３バイト）に書き込み、ブロックＢＬＯＣＫ８７に続れるようであれば、ブロックＢＬＯＣＫ８７のチェーンブロックエリア３５の指示するブロックＢＬＯＣＫの更新カウンタ３１を『１』インクリメントして加算値が、例えば１万回を越えているかどうかを調べ、指示されるブロックＢＬＯＣＫの更新カウンタ３１が１万回を越えるようであれば、更新回数が１万回以下のブロックＢＬＯＣＫを探し当て、そのブロックＢＬＯＣＫの番号を直前に書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５に書き込む。このようにして、データの書き込みが行われ、更新回数が１万回を越えるブロックＥＬＯＣＫが排除されて行く。そして、書き込みデータがなくなるまで同様の操作を行い、最後に書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５に記憶されていた内容を新しい未使用のスタートブロック番号ＯＳＢに書き換え、ポインタブロック１ａの書き換え回数ＷＣＮＴを『１』インクリメントして『１３８９₁₆』となり、最後にデータを書き込んだブロックＢＬＯＣＫのチェーンブロックエリア３５を『ＦＦ₁₆』にする。そして、ディレクトリブロック３０の最終ブロック番号を記憶するエンドブロック番号エリア３４に最後のデータを書き込んだブロックＢＬＯＣＫの番号を書き込むとともに、更新カウンタ３１を『１』インクリメントすると、第５図（ｂ）に示されるように、更新カウンタ３１が『１４３０₁₆』となり、ファイル４のスタートブロック番号エリア３３が『３３₁₆』で、エンドブロック番号エリア３４が『３７₁₆』となる。

次に第５図（ａ），（ｂ）を参照しながらＥＥＰＲＯＭ１に書き込まれているファイル１の削除動作について説明する。

ディレクトリブロック３０となるブロックＢＬＯＣＫ１よりファイル１を探し、ファイル領域３２の先頭の２バイトを『００₁₆』とする。次いで、ディレクトリブロック３０の更新カウンタ３１を『１』インクリメントし、ファイル１のスタートブロック番号エリア３３とエンドブロック番号エリア３４のデータを参照して、ポインタブロック１ａのエンドブロック番号ＯＥＢが指示するブロックのチェーンブロックエリア３５の内容（削除直前までは『ＦＦ₁₆』であった）をスタートブロック番号エリア３３の内容に変更し、このブロックの更新カウンタ３１を『１』インクリメントする。すなわち、未使用ブロックの最後に今削除したファイル４を接続するわけである。このようにして、更新カウンタ３１を進めながら何度もファイルの更新，削除を実行して行くうちに、更新カウンタ３１が１万回に接近する。

次に更新カウンタ３１が１万回に到達した場合のアクセス処理について説明する。

まず、ポインタブロック1aのスタートブロック番号ＯＳＢの内容が示しているブロックＢＬＯＣＫのチェーンブロックエリア３５の内容を新規のスタートブロック番号ＯＳＢとする。次いで、このブロック直前のディレクトリブロック３０の更新カウンタ３１の情報以外の内容を転送する。そして、ポインタブロック1aのディレクトリＤＢに新規のディレクトリブロック番号を書き込み、ポインタブロック1aの書き換え回数ＷＣＮＴおよび更新カウンタ３１を『１』インクリメントする。

一方、ポインタブロック1aの書き換え回数ＷＣＮＴは１万回を越えた場合は、予備ポインタブロックＳＰＢ１～ＳＰＢ５０のうち一番近い予備ポインタブロックへ書き換え回数ＷＣＮＴの情報以外のデータを転送し、新規のポインタブロックの書き換え回数ＷＣＮＴ（$0000_{16}$）を『１』インクリメントして『$0001_{16}$』に設定する。この場合、破棄されたポインタブロック1aの書き換え回数ＷＣＮＴは１万回以上となり、新のポインタブロック1aの書き換え回数ＷＣＮＴは１万回以下となる。このようにして、ディレクトリブロック３０およびポインタブロック1aの書き込み削除を管理する。また削除されたファイルが使用していたブロックは未使用ブロックの一番最後に回される。これは、未使用ブロックの使用回数を平均化するためである。しかし、使用されているファイルが更新されずにずっとそのままであると、そのファイルが使用しているブロックは更新回数がそのまま変化しない。例えば、最初に作成されたファイルがそのままずっと登録されたまま残っていると、他のブロックは更新回数が５００回以上なのに、このファイルだけは２回というようなアンバランスが生じる。そこで、ＥＥＰＲＯＭ１の使用状態を平均化するための補正処理を行う。

補正処理起動条件は下記（ａ），（ｂ）の場合においてである。

（ａ）ポインタブロック1aの書き換え回数ＷＣＮＴの値が２５６の整数倍になった時点。

（ｂ）ファイルを構成するブロックの更新カウンタ３１の平均値が一番低い値と、未使用ブロックの更新カウンタ３１の平均値との差が２５６を越えた時点。

すなわち、新規のファイルを作成したり、削除したりした後、ポインタブロック1aの書き換え回数ＷＣＮＴを見て、ちょうど２５６の整数倍、２バイトの１６進数の下桁が『$00_{16}$』になった時点で、ディレクトリブロック３０に登録されている順にファイルを検査して行く。そして、ファイルを構成するブロックの更新カウンタ３１を加算し、さらに構成するブロック数で除して更新平均値を算出する。次いで、次のファイルに対しても同様の更新平均値を出して比較し、低いものをその比較対照として残し、次のファイルの更新平均値と比較して行き、一番低い更新回数で構成されるファイルを探し出す。そして、未使用のブロックの更新回数の平均を計算し、平均更新回数の一番低いファイルとの差を算出する。

次に第６図（ａ）～（ｃ）を参照しながら補正処理動作について説明する。

第６図（ａ）～（ｃ）はこの発明による補正動作を説明する模式図であり、これらの図において、４１はファイルで、ブロックＢＬＯＣＫ５～７で構成され、平均更新回数が最も低いものである。４２は未使用ブロック群で、ポインタブロック1aの未使用スタートブロック番号ＯＳＢで指示される。未使用ブロック群４２は、ブロックＢＬＯＣＫ５０，１０，１１，１８，５５，８０，８１が１つのチェイン構造となっている。

まず、ファイル４１のスタートブロックを未使用ブロック群４２のブロックＢＬＯＣＫ８１の後段に接続させるため、同図（ａ）に示すようにブロックＢＬＯＣＫ８１のチェーンブロックエリア３５の内容が『$FF_{16}$』から同図（ｂ）に示すように、チェーンブロックエリア３５の内容が『$05_{16}$』、すなわち、ファイル４１のブロックＢＬＯＣＫ５を指示させ、さらに、ファイル４１のブロックＢＬＯＣＫ５～７（ファイル４１のチェーンブロックエリア３５の内容が『$FF_{16}$』になる

まで）を未使用ブロック群４２のブロックＢＬＯＣＫ８１の後段に接続させ、同図（ｃ）に示すように、ディレクトリブロック３０のファイル４１のスタートポインタをブロックＢＬＯＣＫ５０にするとともに、終了ポインタをブロックＢＬＯＣＫ１１に変更する。次いで、未使用ブロック群４２の未使用のスタートブロック番号ＯＳＢを『１２₁₆』、未使用のエンドブロック番号ＯＥＢを『０Ｂ₁₆』にする。これにより、登録されたファイルを構成するブロックに対しても再度使用可能となり、ＥＥＰＲＯＭ１全体のブロックが平均的に使用、更新されることになる。第７図は第１図（ａ）に示したＥＥＰＲＯＭ１のデータ書き込み制御動作を説明するためのフローチャートである。なお、(1)～(18)は各ステップを示す。

まず、ディレクトリブロック３０の空エリアを探して、新規のファイル名を書き込む(1)。次いで、未使用のスタートブロック番号ＯＳＢをＣＰＵ１１のアキュムレータＡＣＣに記憶させる(2)。アキュムレータＡＣＣが指示するブロックの書き換え回数ＷＣＮＴを＋１更新する(3)。ここで、書き換え回数ＷＣＮＴが１００００を越えたかどうかを判断し(4)、ＹＥＳならばアキュムレータＡＣＣの指示するブロックの冗長ブロックエリアＣＢをアキュムレータＡＣＣに記憶し(5)、ステップ(3)に戻り、ＮＯならばディレクトリブロック３０のスタートブロック番号エリア（ＳＢ）３３にアキュムレータＡＣＣの内容を書き込む(6)。次いで、アキュムレータＡＣＣが指示するブロックのデータエリアにデータを書き込む(7)。ここで、書き込みデータがアキュムレータＡＣＣが指示するブロックの容量が２３５バイトを越えるかどうかを判断し(8)、ＹＥＳならばアキュムレータＡＣＣが指示するブロックの冗長ブロックエリアＣＢをアキュムレータＢＣＣに記憶させる(9)。次いで、アキュムレータＢＣＣが指示するブロックの書き換え回数ＷＣＮＴを＋１更新する(10)。次いで、書き換え回数ＷＣＮＴが１００００を越えたかどうかを判断し(11)、ＹＥＳならばアキュムレータＢＣＣの指示するブロックの冗長ブロックエリアＣＢを記憶させ(12)、ステップ(10)に戻り、ＮＯならばアキュムレータＡＣＣが指示するブロックの冗長ブロックエリアＣＢにアキュムレータＢＣＣの内容を書き込み(13)、ステップ(7)に戻る。

一方、ステップ(8)の判断でＮＯの場合は、アキュムレータＡＣＣが指示する冗長ブロックエリアＣＢを未使用のスタートブロック番号ＯＳＢに書き込む(14)。次いで、ポインタブロック１ａの書き換え回数ＷＣＮＴを＋１更新する(15)。次いで、アキュムレータＡＣＣが指示するブロックの冗長ブロックエリアＣＢへ「ＦＦ₁₆」を書き込む(16)。そして、ディレクトリブロック３０の新ファイル位置のエンドブロック番号エリア３４へアキュムレータＡＣＣの内容を書き込む(17)。次いで、ディレクトリブロック３０の書き換え回数ＷＣＮＴを更新する(18)。

第８図はこの発明による初正期制動作手順を説明するためのフローチャートである。なお、(1)～(7)は各ステップを示す。

ポインタブロック１ａの書き換え回数ＷＣＮＴが２５６の整数倍であるかどうかを判断し(1)、ＮＯならばリターン（ＲＥＴＵＲＮ）し、ＹＥＳならばディレクトリブロック３０に登録された各ファイルを構成するブロックの更新カウンタ３１の平均値を算出して、最も更新回数が少ないファイルを探し出す(2)。次いで、未使用のブロックの更新回数の平均値を算出する(3)。次いで、未使用ブロックの更新カウンタの平均値からファイルを構成するブロックの更新カウンタの平均値の最小値を減算し、さらに減算値から２５６を差し引いた値が正かどうかを判断し(4)、ＮＯならばリターンし、ＹＥＳならば未使用ブロックの最後尾に該当するファイルのヘッドを接続させる(5)。次いで、接続したファイルの内容を未使用ブロックへ転送させ(6)、ディレクトリブロック３０にある接続したファイルのスタートポインタ、エンドポインタを変更し(7)、リターンする。

〔発明の効果〕

以上説明したように、この発明は記憶領域を複数に分割し、各ブロック毎に書き込み回数を記憶し、この書き込み回数に応じて書き込み頻度の低いブロックを未使用ブロックの先頭から書き込むとともに、書き込み頻度の低いブロックを未使用ブロックの最後尾に接続させるようにしたので、EEPROMに書き込まれるデータの消失を防止するとともに、EEPROMへの書き込み回数を平均化させることができる。またEEPROM上の各ブロックの書き換え頻度を平均化でき、書き換え寿命を大幅に延長できる優れた利点を有する。

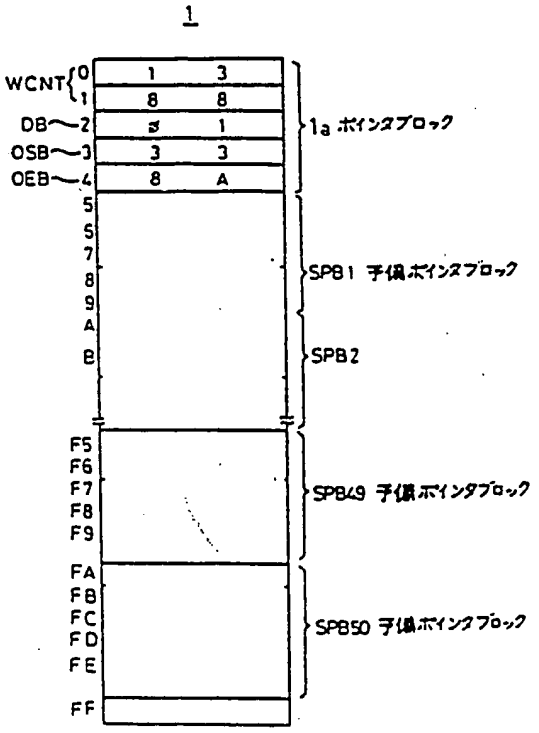
4．図面の簡単な説明

第1図（a）はこの発明の一実施例を示すプログラマブルリードオンリメモリへの書き込み回数管理方式を説明する模式図、第1図（b）はこの発明の装置構成を説明するためのブロック図、第2図は第1図（a）に示すEEPROMの構造を示す模式図、第3図は第2図に示すディレクトリブロック構造を説明する模式図、第4図は未使用のEEPROMの状態を説明する模式図、第5図（a），（b）はEEPROMへの書き込み動作を説明する模式図、第6図（a）～（c）はこの発明による補正処理動作を説明する模式図、第7図は第1図（a）に示したEEPROMのデータ書き込み制御動作を説明するためのフローチャート、第8図はこの発明による補正制御動作手順を説明するためのフローチャートである。

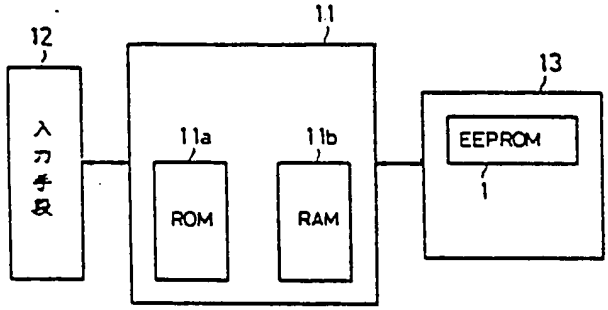
図中、1はEEPROM、1aはポインタブロック、21はブロック番号、30はディレクトリブロック、31は更新カウンタ、32はファイル領域、33はスタートブロック番号エリア、34はエンドブロック番号エリア、35はチェーンブロックエリア、41はファイル、42は未使用ブロック群である。

代理人　小　林　将　高

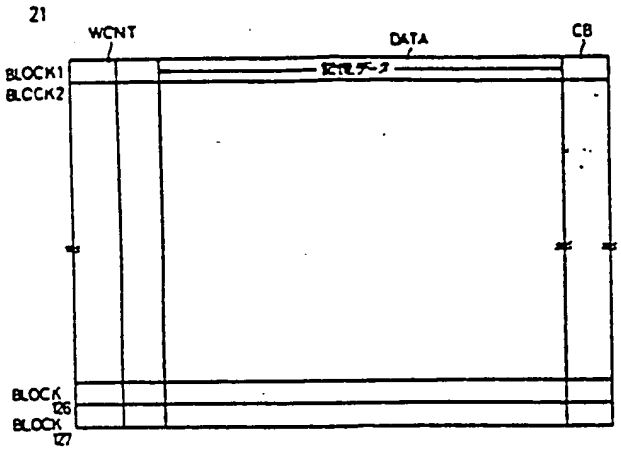
第　1　図　(a)



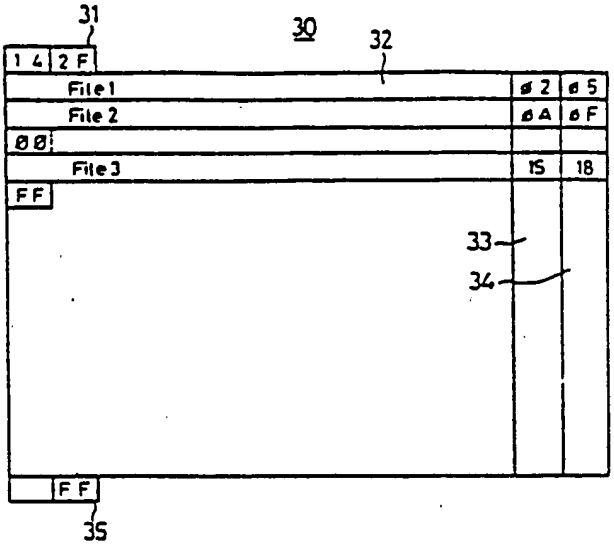

第　1　図　(b)



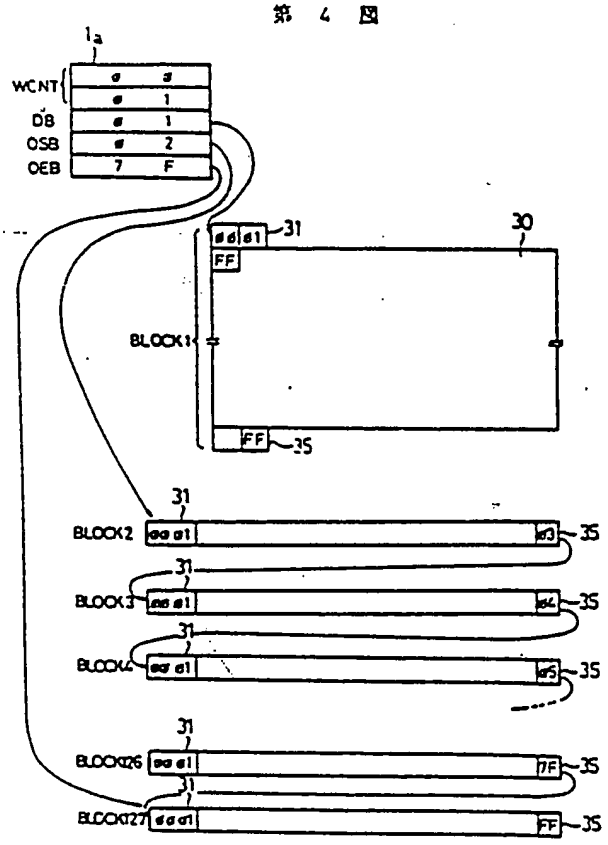
第 2 図



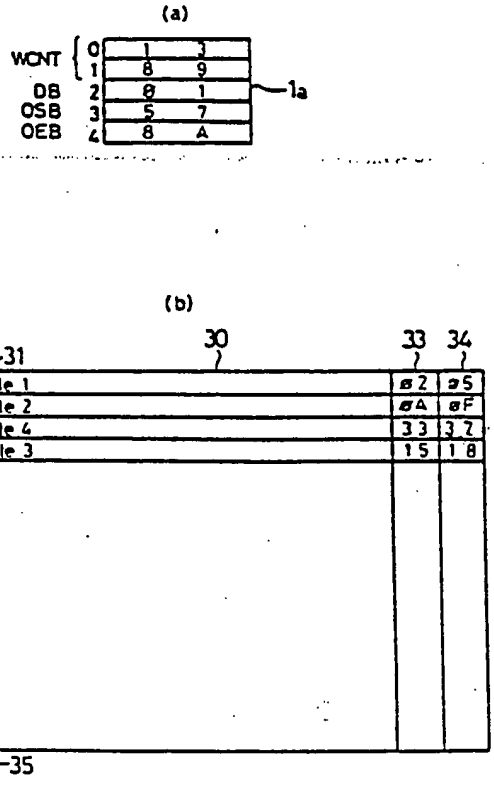

第 3 図



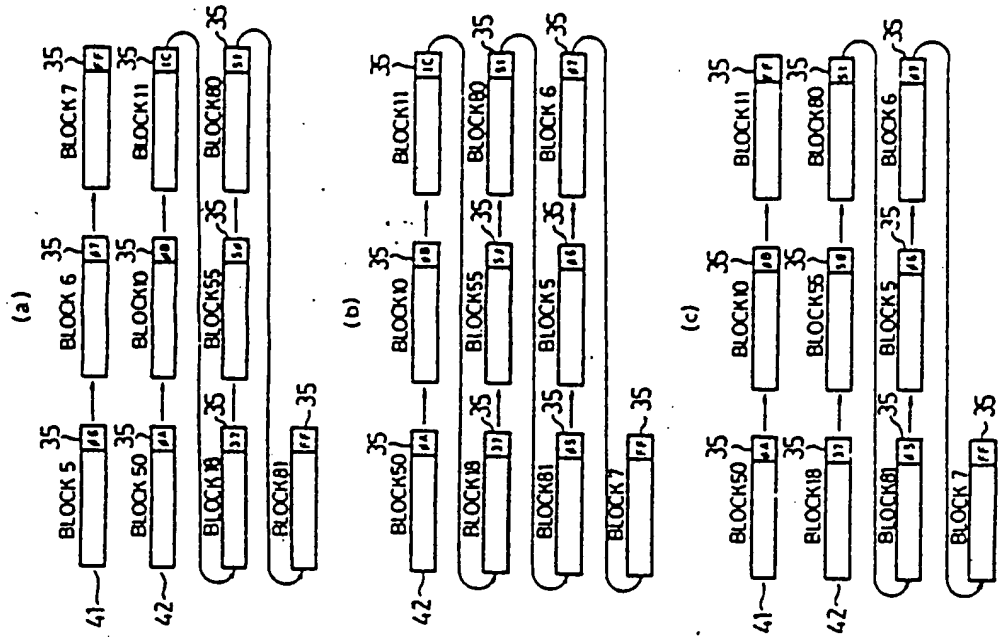

第 4 図



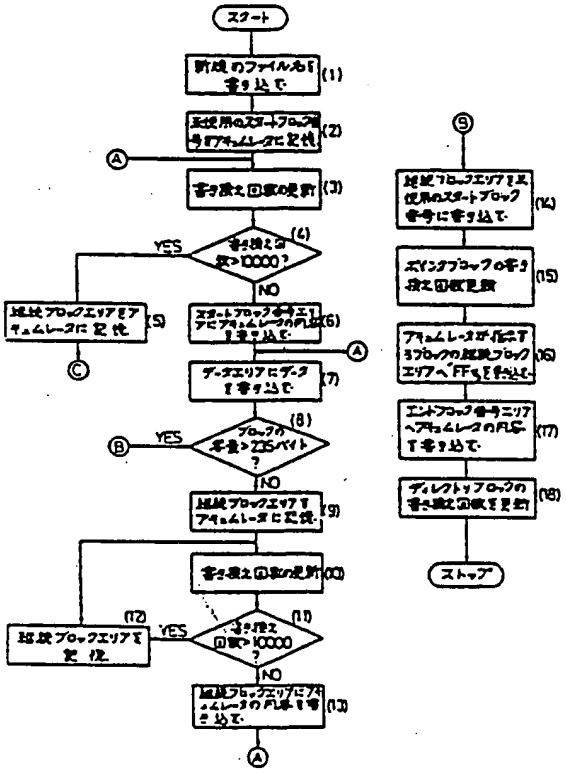

第 5 図



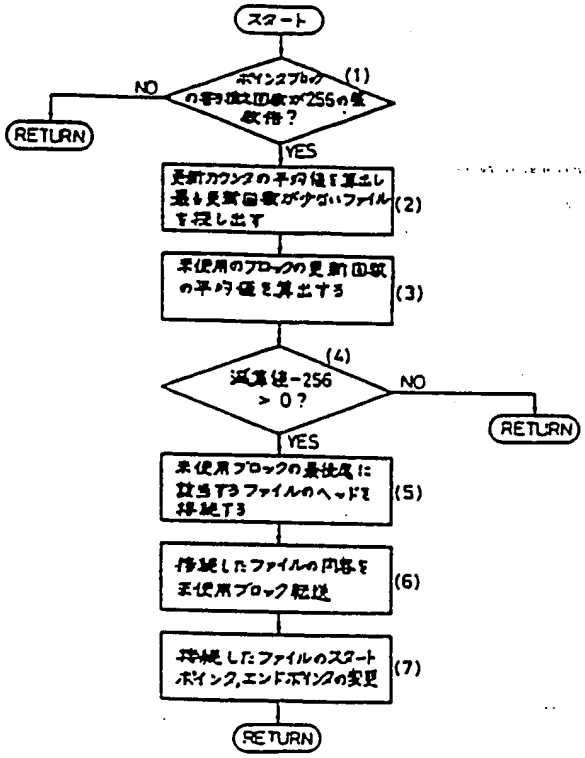
第 6 図


(a)
BLOCK 7
BLOCK 11
BLOCK 80
BLOCK 6
BLOCK 10
BLOCK 55
BLOCK 5
BLOCK 50
BLOCK 18
BLOCK 81
41
42
(b)
BLOCK 11
BLOCK 80
BLOCK 6
BLOCK 10
BLOCK 55
BLOCK 5
BLOCK 50
BLOCK 18
BLOCK 81
BLOCK 7
42
(c)
BLOCK 11
BLOCK 80
BLOCK 6
BLOCK 10
BLOCK 55
BLOCK 5
BLOCK 50
BLOCK 18
BLOCK 81
BLOCK 7
41
42

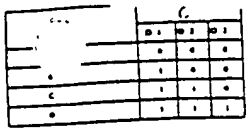
第 7 図


スタート
(1)
(2)
(3)
(4)
YES
NO
(5)
(6)
(7)
(8)
YES
NO
(9)
(10)
(11)
YES
NO
(12)
(13)
(14)
(15)
(16)
(17)
(18)
ストップ

第 8 図

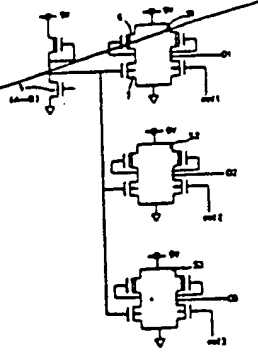

スタート
(1)
NO
YES
RETURN
(2)
(3)
(4)
NO
YES
RETURN
(5)
(6)
(7)
RETURN

...61-126835 (22) 30.5.1986
...LTD (72) HIROHIKO SATO(1)
Cl⁴. G11C17/00,H01L27/10



:POSE: To improve the integrated degree of a ROM by making the threshold ... of a MOS transistor more than four types and applying the memory ...y of more than 2 bits by one transistor.

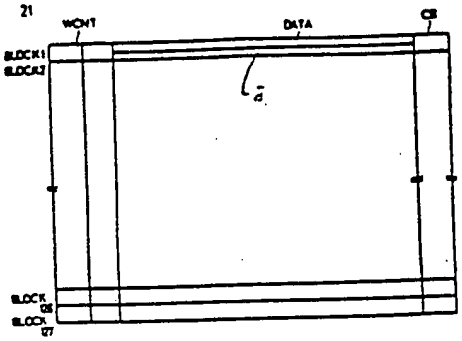
:SOLUTION: The threshold voltage of the MOS transistor 5 of a memory transistor is made more than four types of the voltages A~D and the transistor 5 is detected by three types of detecting circuits S1~S3 forming the differential amplifier of reference voltages ref1~ref3 corresponding to the respective thresholds A~D. Then, the detected outputs D1~D3 of the circuits S1~S3 indicate the values of a table I, when a logical processing is applied to combine more than two types of the outputs D1~D3, the capacity of more than two bits is applied to the one memory transistor. Consequently, the number of the memory transistors can be reduced and the integrated degree of the ROM can be improved.



a: table I, b: memory transistor, c: output

AL

) MANAGEMENT SYSTEM FOR OF NUMBER OF TIMES OF WRITING PROGRAMMABLE READ ONLY MEMORY
) 62-283496 (A) (43) 9.12.1987 (19) JP
) Appl. No. 61-124731 (22) 31.5.1986
) CANON INC (72) SHINICHI NAKADA
) Int. Cl⁴. G11C17/00

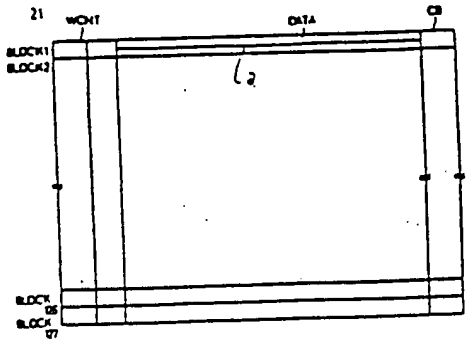



JRPOSE: To prolong the rewriting life of an EEPROM by connecting a block having little frequency in writing in a divided memory area to an unused block.

)NSTITUTION: The EEPROM in which erasing, rewriting or the like are controlled through an input means, a CPU or the like is divided into 127 such ...pointer blocks BLOCK1~BLOCK127. The respective blocks include an area ...NT for storing the number of times of updating the block, a memory data ...a DATA, a continuous block area CB corresponding to the continuous block number when the memory data is supplied to other block or the absence of the continuous block number when the data is not supplied to other block or the like. Then, the area WCNT is referred through the CPU, the block of little frequency in the writing is connected to the last of the unused block to rewrite the CB. Accordingly, the respective blocks are averaged and used and the writing life of the EEPROM used for a card or the like is prolonged.

a: storage data

54) MANAGEMENT SYSTEM FOR OF NUMBER OF TIMES OF WRITING PROGRAMMABLE READ ONLY MEMORY
11) 62-283497 (A) (43) 9.12.1987 (19) JP
21) Appl. No. 61-124732 (22) 31.5.1986
71) CANON INC (72) SHINICHI NAKADA
51) Int. Cl⁴. G11C17/00




PURPOSE: To average the rewriting frequency of a memory block and to prolong the life of an EEPROM by suppressing the writing to the memory block reaching the number of times of setting.

CONSTITUTION: The memory area of the EEPROM in which the erasing, the rewriting or the like are carried out by an input means, a CPU or the like is divided into 127 such as blocks BLOCK1~BLOCK127 and in the respective blocks, an area WCNT for storing the updating and rewriting number of times as well as a data memory area DATA is provided. When the contents of the ... WCNT in which the counted value of the updating counter of the directory ... of a block pointer is written are referred to and reach the set number, ...e rewriting of the block is suppressed through the CPU. The rewriting frequency of the respective blocks is averaged and the life of the EEPROM is prolonged.

a: storage data